Roteiro

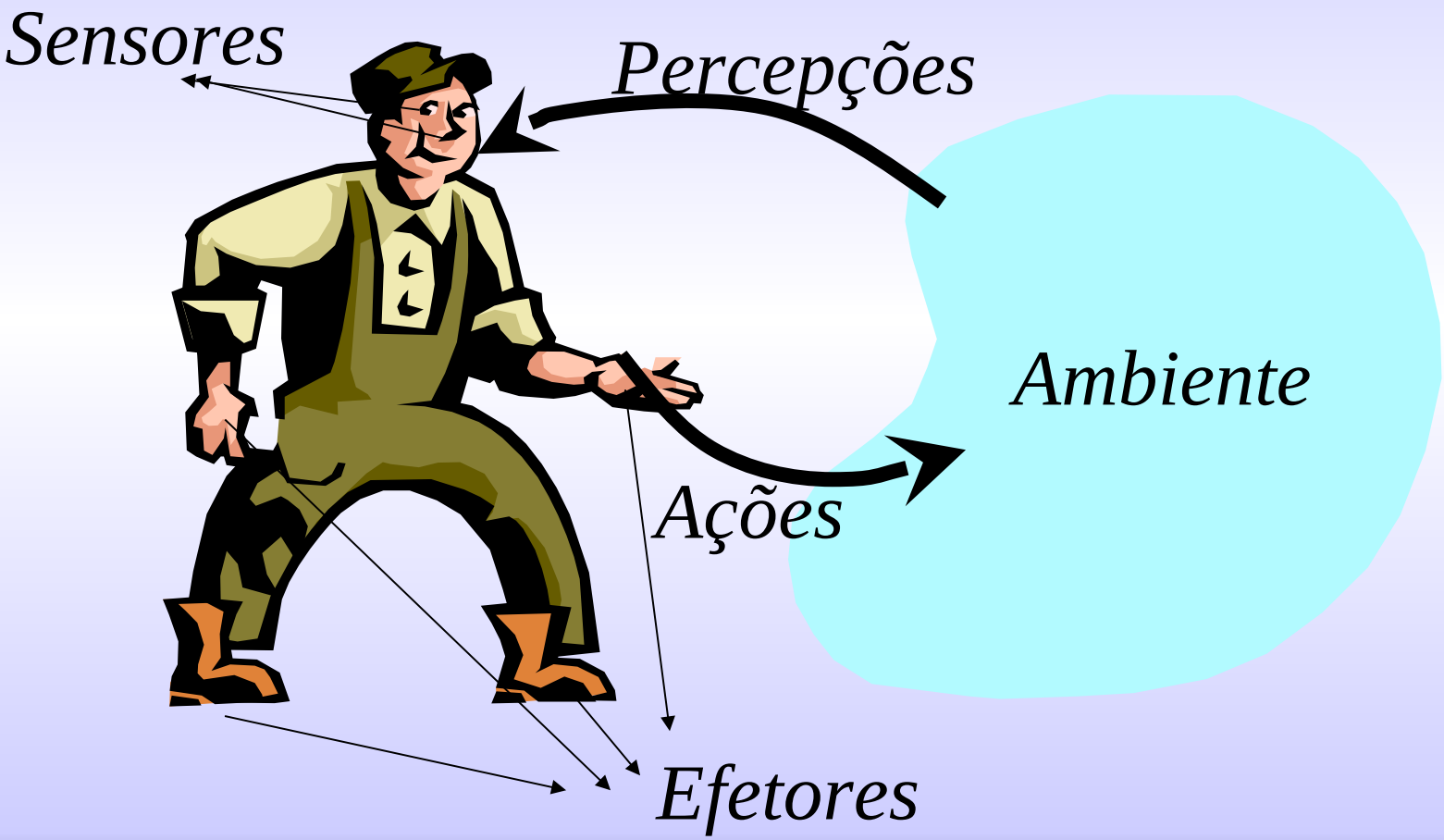
- Introdução
- Como os agentes deveriam agir
  - O mapeamento ideal das seqüências de percepção às ações
  - Autonomia
- Estrutura de Agentes Inteligentes
  - PAGE
  - Programas Agentes
  - Por que simplesmente não olhamos as respostas?
  - Um exemplo
  - Agentes de Reativo Simples
  - Agentes que observam o mundo (baseados em modelos)
  - Agentes baseados em objetivos
  - Agentes baseados em utilidade
- Ambientes
  - Propriedades
  - Programas

Introdução

Um ***agente*** é qualquer coisa que pode ***perceber*** seu ambiente por ***sensores*** e ***agir*** sobre seu ambiente por ***efetores***.

Como os agentes deveriam agir?

- Um agente racional
  - FAZ A COISA CERTA



Como os agentes deveriam agir?

- Medida de desempenho
  - critério que determina o sucesso do agente
- Oniciência é diferente de racionalidade
- Um agente racional depende:
  - da medida de desempenho que define o grau de sucesso
  - da seqüência de percepção
  - do que o agente sabe sobre o ambiente
  - das ações que o agente pode realizar
- Um agente racional ideal
  - Para cada seqüência de percepção possível, um **agente racional ideal** deve saber se sua ação maximizará sua medida de desempenho, baseado na evidência de sua seqüência de percepção e no conhecimento que o agente traz consigo.

O mapeamento ideal

- mapeamento
  - uma lista muito longa (na verdade infinita, mas limitada para o mundo real) de seqüências de percepção sendo considerada
- mapeamentos ideais
  - especifica qual ação um agente deve tomar em resposta a qualquer seqüência de percepção
- exemplo:
  - a função raiz quadrada
  - a seqüência de percepção: a digitação de números
  - a ação: mostrar o resultado correto

O mapeamento ideal

Percepção (x)	Ação z = SQRT(x)
1.0	1.000000000000000
1.1	1.048808848170152
1.2	1.095445115010332
1.3	1.140175425099138
1.4	1.183215956619923
1.5	1.224744871391589
1.6	1.264911064067352
1.7	1.303840481040530
1.8	1.341640786499874
1.9	1.378404875209022
....	....

function SQRT(x)

$z \leftarrow 1.0$

repeat

$z \leftarrow z - (z^2 - x)/(2z)$

until $|z^2 - x| < 10^{-15}$

return z

Autonomia

- Conhecimento Interno
  - sistemas baseados somente no conhecimento interno e sem atenção à sua percepção não são autônomos (são pré-cientes)

Um sistema é autônomo quando seu ***comportamento*** *é determinado pela sua* ***experiência***.

- Um agente verdadeiramente autônomo
  - é capaz de operar em uma ampla diversidade de ambientes
  - dado que houve tempo suficiente para se adaptar

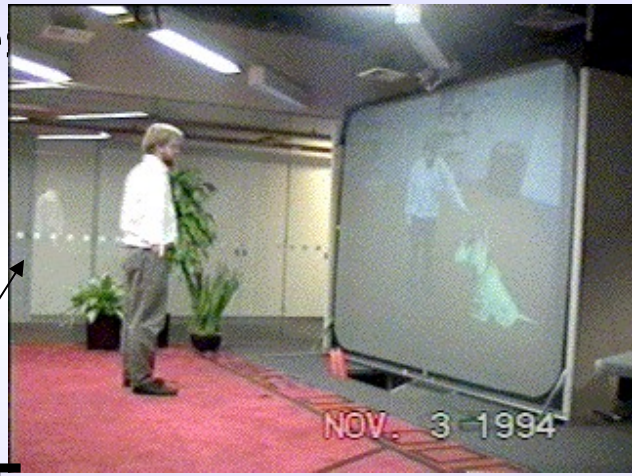
Estrutura de Agentes Inteligentes

ambiente real

- O trabalho da IA
  - fazer um
    - Agente = Arquitetura + Programa
  - ambientes híbridos?

softbots

ALIVE



Programas Agentes

function ESQUELETO_DO_AGENTE(percepção): ação
    static: memória (memória do agente sobre o mundo)

    memória ← ATUALIZA_MEMÓRIA(memória, percepção)
    ação ← ESCOLHA_A_MELHOR_AÇÃO(memória)
    memória ←ATUALIZA_MEMÓRIA(memória, ação)

    return ação

Por que simplesmente não olhamos as respostas?

function AGENTE_POR_TABELA(percepção): ação

static: *percepções* (uma seqüência, inicialmente vazia)
tabela (uma tabela, indexada por percepção,
inicialmente completamente especificada)

agregar (percepção) ao fim de *percepções*
ação ← LOOKUP(percepções, tabela)

return ação

- Observação:
  - Tabela para jogar xadrez = 35^{100} entradas

Perceptions. Actions. Goals. Environment.

- Deve-se primeiro especificar o cenário para o projeto de agente inteligente
- Considere, por exemplo, o projeto de um táxi automático.
  - Quais são as/os
    - Percepções?
    - Ações?
    - Objetivos?
    - Ambiente?

P. A. G. E.

- Deve-se primeiro especificar o cenário para o projeto de agente inteligente

- Considere, por exemplo, o projeto de um táxi automático.
  - Quais são as/os
    - Percepções?
      - vídeo, acelerômetros, ponteiros, sensores no motor, teclado, GPS, ...
    - Ações?
      - girar o volante, acelerar, frear, buzinar, falar / imprimir
    - Objetivos?
      - segurança, chegar ao destino, maximizar lucros, obedecer as leis, conforto do passageiro...
    - Ambiente?
      - ruas urbanas das metrópoles brasileiras, rodovias duplicadas, tráfego, pedestres, condições meteorológicas, consumidores

P. A. G. E.

- Deve primeiro especificar o cenário para o projeto de agente inteligente
- Considere, por exemplo, o projeto de um agente de compras pela internet.
  - Quais são as/os
    - Percepções?
    - Ações?
    - Objetivos?
    - Ambiente?

Propriedades dos ambientes

- COMPLETAMENTE OBSERVÁVEL x PARCIALMENTE OBSERVÁVEL
  - quanto ao acesso completo do estado do ambiente fornecido pelo aparato sensorial
  - Um agente aspirador de pó com apenas um sensor não poderia saber se está sujo o local B

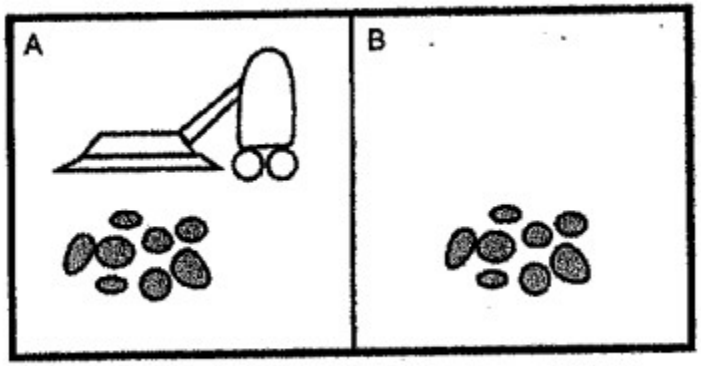


Figura 2.2 Um mundo de aspirador de pó com apenas dois locais.

Propriedades dos ambientes

- DETERMINÍSTICO x ESTOCÁSTICO
  - próximo estado do ambiente é completamente determinado pelo estado corrente e pelas ações selecionadas pelo agente.
  - Lembra um AFD
  - Um motorista de táxi é ESTOCÁSTICO, pois não se pode prever o comportamento do tráfego, ou se irá furar um pneu.

- EPISÓDICO x SEQUENCIAL
  - a experiência do agente é dividida em episódios
  - cada episódio = percepção + ação
  - em uma linha de montagem, um agente localizador de peças defeituosas baseia cada decisão na peça atual, não importa se anteriores estavam ou não defeituosas. A decisão atual não afeta a próxima peça (EPISÓDICO)
  - uma decisão no jogo de xadrez afeta os próximos movimentos (SEQUENCIAL)

Propriedades dos ambientes

- ESTÁTICO versus DINÂMICO
  - o ambiente pode se alterar enquanto um agente está decidindo
  - se o ambiente não se altera, mas o desempenho do agente se altera é chamado SEMIDINÂMICO
- DISCRETO versus CONTÍNUO
  - percepções e ações são contáveis e distintas (xadrez é discreto, táxi é contínuo)
  - no ambiente CONTÍNUO ocorrem variações ao longo do tempo.

- AGENTE ÚNICO x MULTIAGENTES
  - um agente que resolve palavras cruzadas está claro que é único
  - um agente que joga xadrez está competindo com outro agente.
  - um agente B de xadrez tenta maximizar o seu desempenho e com isso minimiza o desempenho do agente B, sendo então um sistema multiagente competitivo
  - Veja implementações de vários ambientes em aima.cs.berkeley.edu

Tipos de Ambientes

	Paciência	Gamão	Compras na Internet	Táxi
Observável				
Determinístico				
Episódico				
Estático				
Discreto				

Tipos de Ambientes

	Paciência	Gamão	Compras na Internet	Táxi
Observável	SIM	SIM	NÃO	NÃO
Determinístico	SIM	NÃO	PARCIAL	NÃO
Episódico	NÃO	NÃO	NÃO	NÃO
Estático	SIM	SEMI	SEMI	NÃO
Discreto	SIM	SIM	SIM	NÃO

- O tipo de ambiente determina fortemente o projeto do agente.
- O mundo real é:
  - » INACESSÍVEL (parcialmente observavel)
  - » ESTOCÁSTICO (não determinístico)
  - » SEQÜENCIAL (não episódico)
  - » DINÂMICO (não estático)
  - » CONTÍNUO (não discreto)

Tipos de Agentes

- Quatro tipos básicos em ordem crescente de generalidade:

  - AGENTES REATIVOS SIMPLES
  - AGENTE DE REFLEXO COM ESTADOS
  - AGENTES BASEADOS EM OBJETIVOS
  - AGENTES BASEADOS EM UTILIDADE

Baseado em modelos

Tipos de Agentes

- AGENTES REATIVO SIMPLES
  - Seleciona ações conforme a percepção atual

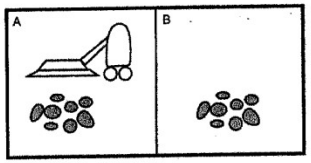


Figura 2.2 Um mundo de aspirador de pó com apenas dois locais.

Seqüência de percepções	Ação
[A, Limpo]	Direita
[A, Sujo]	Aspirar
[B, Limpo]	Esquerda
[B, Sujo]	Aspirar
[A, Limpo], [A, Limpo]	Direita
[A, Limpo], [A, Sujo]	Aspirar
⋮	⋮
[A, Limpo], [A, Limpo], [A, Limpo]	Direita
[A, Limpo], [A, Limpo], [A, Sujo]	Aspirar
⋮	⋮

Figura 2.3 Tabulação parcial de uma função de agente simples correspondente ao mundo de aspirador de pó mostrado na Figura 2.2.

função AGENTE-ASPIRADOR-DE-PÓ-REATIVO([*posição*,*estado*]) **retorna** uma ação*

se *estado*= *Sujo* **então retorna** *Aspirar*
senão se *posição* = *A* **então retorna** *Direita*
senão se *posição* = *B* **então retorna** *Esquerda*

- E se o aspirador não tivesse sensor de posição?

O exemplo do táxi

- considerando a tabela
  - 25 quadros por segundo
  - cada imagem = 1000x1000 pixels com 8 bits de cor e 8 bits de intensidade
  - ou seja, 50Mbytes por segundo
  - para mapear todas as possibilidades de uma corrida de uma hora teríamos que ter...

    60 minutos $\times$ 60 segundos $\times$ 50 Mbytes
  - para representar um estado.
  - teríamos então

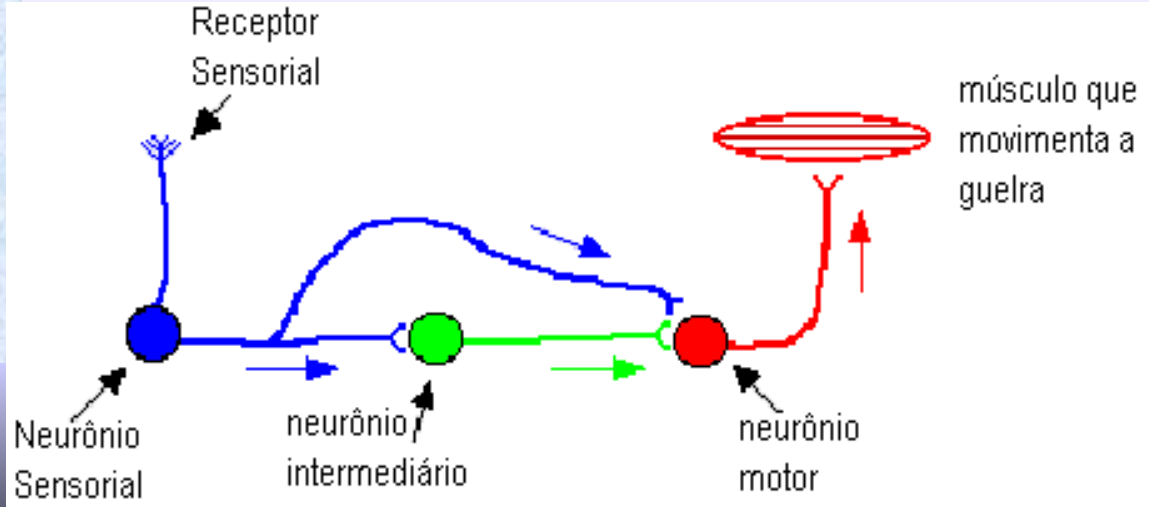
    $2^{60\times60\times400.000.000}$ percepções possíveis
  - Impraticável!!!!!!

O exemplo do táxi

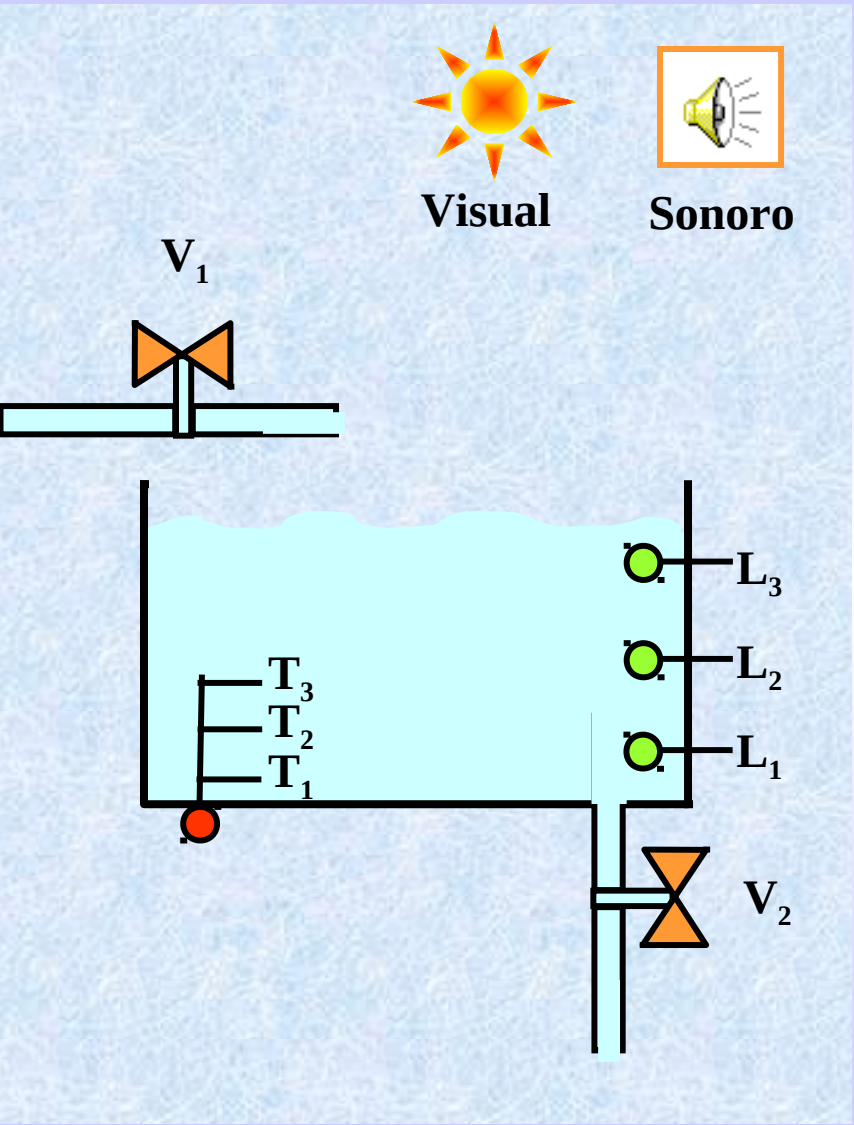
- considerando ocorrências freqüentes e associando entrada e saída

if carro_da_frente_estiver_freando
então inicie_a_freagem

Na natureza



O exemplo da caixa d'água



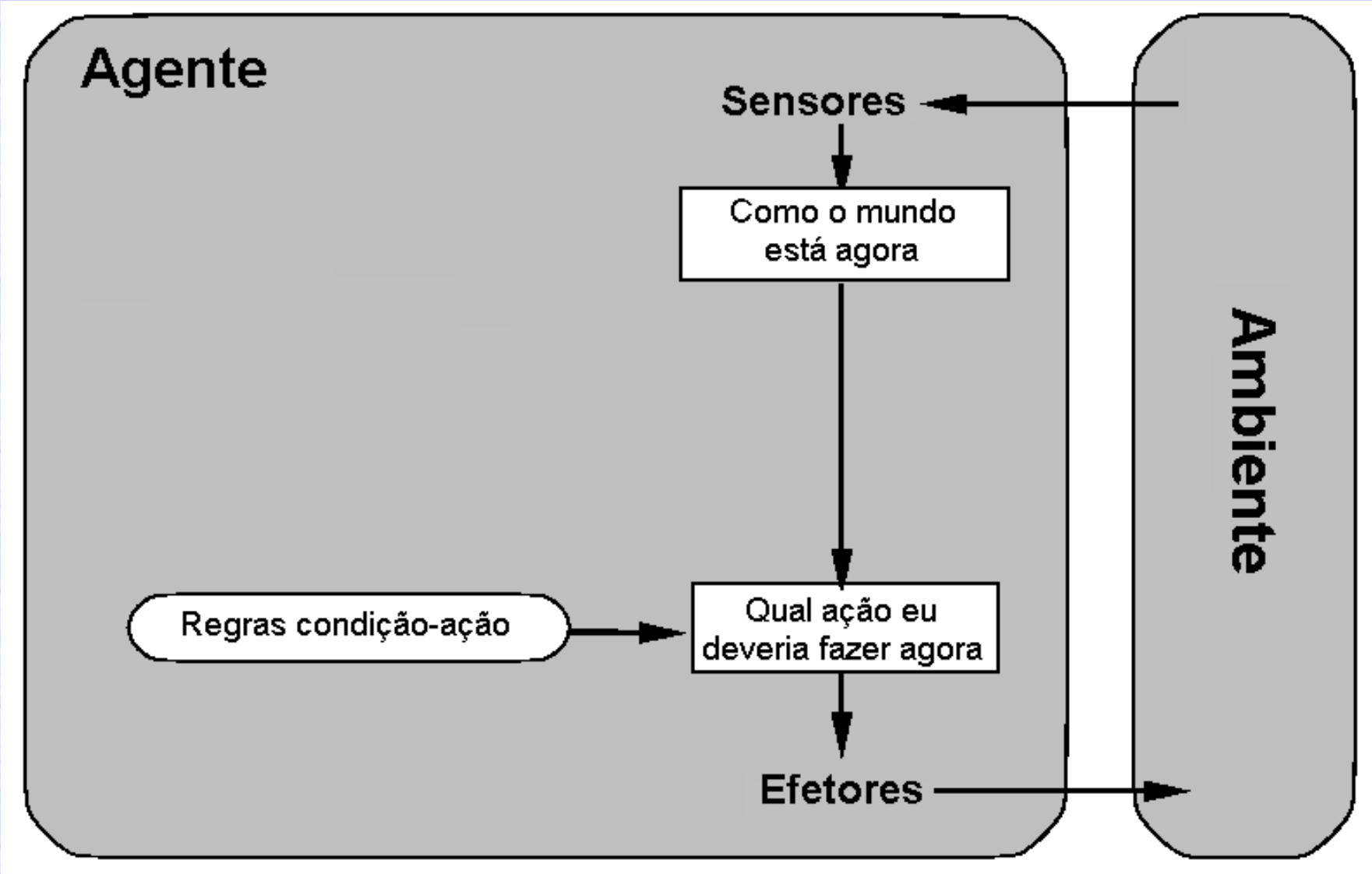
- **Percepções (l e t) :**
  - $t_1 = t < T_1$
    - todos desligados
  - $t_2 = T_1 \leq t < T_2$
    - apenas T_1 ligado
  - $t_3 = T_2 \leq t < T_3$
    - apenas T_1 e T_2 ligados
  - $t_4 = T_3 \leq t$
    - todos ligados

  - $l_1 = l < L_1$
  - $l_2 = L_1 \leq l < L_2$
  - $l_3 = L_2 \leq l < L_3$
  - $l_4 = L_3 \leq l$

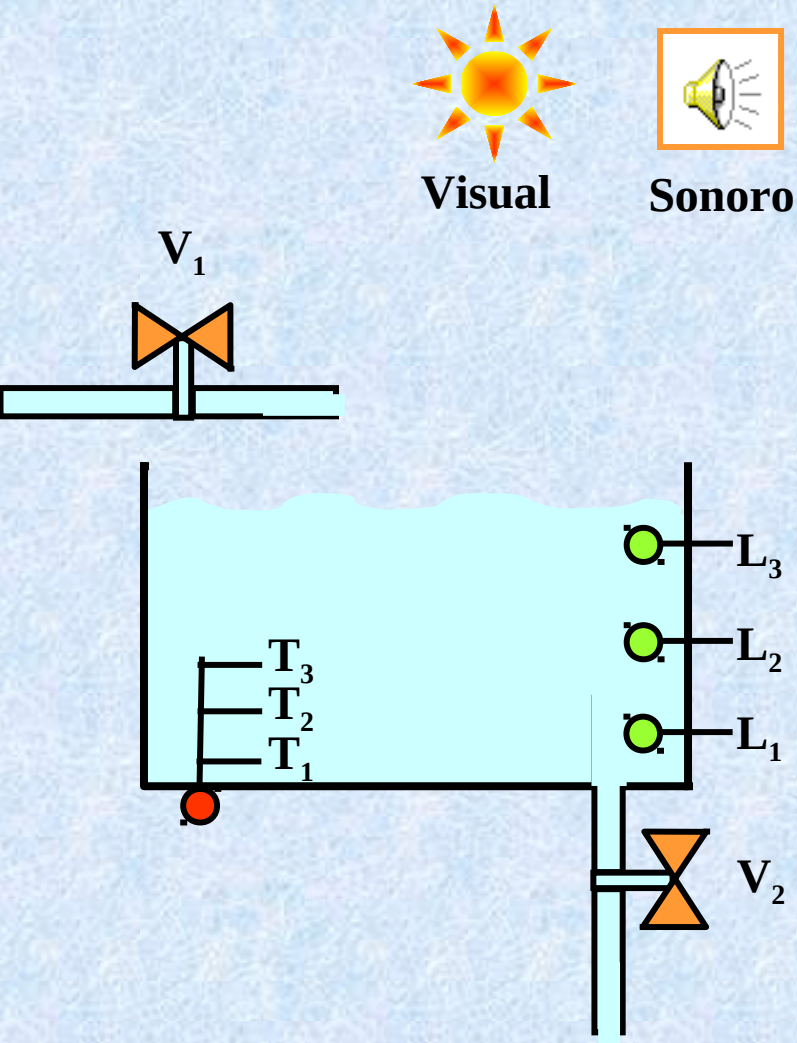
- $2^4 = 16$ possíveis combinações

- **Efetores**
  - V_1, V_2, Alarme Visual e Sonoro

Agente Reativo Simples



O exemplo da caixa d'água



	V_1	V_2	Visual	Sonoro
t_1 l_1	A	F	↓	↓
t_1 l_2	A	F	↓	↓
t_1 l_3	F	A	↓	↓
t_1 l_4	F	A	↑	↓
t_2 l_1	A	F	↓	↓
t_2 l_2	A	F	↓	↓
t_2 l_3	F	A	↓	↓
t_2 l_4	F	A	↑	↓
t_3 l_1	A	F	↓	↓
t_3 l_2	A	F	↓	↓
t_3 l_3	A	F	↑	↓
t_3 l_4	F	F	↑	↑
t_4 l_1	A	F	↑	↓
t_4 l_2	A	F	↑	↓

Agente Reativo Simples

function AGENTE_REATIVO_SIMPLES(percepção): ação

static:

regras (um conjunto regra-ação)

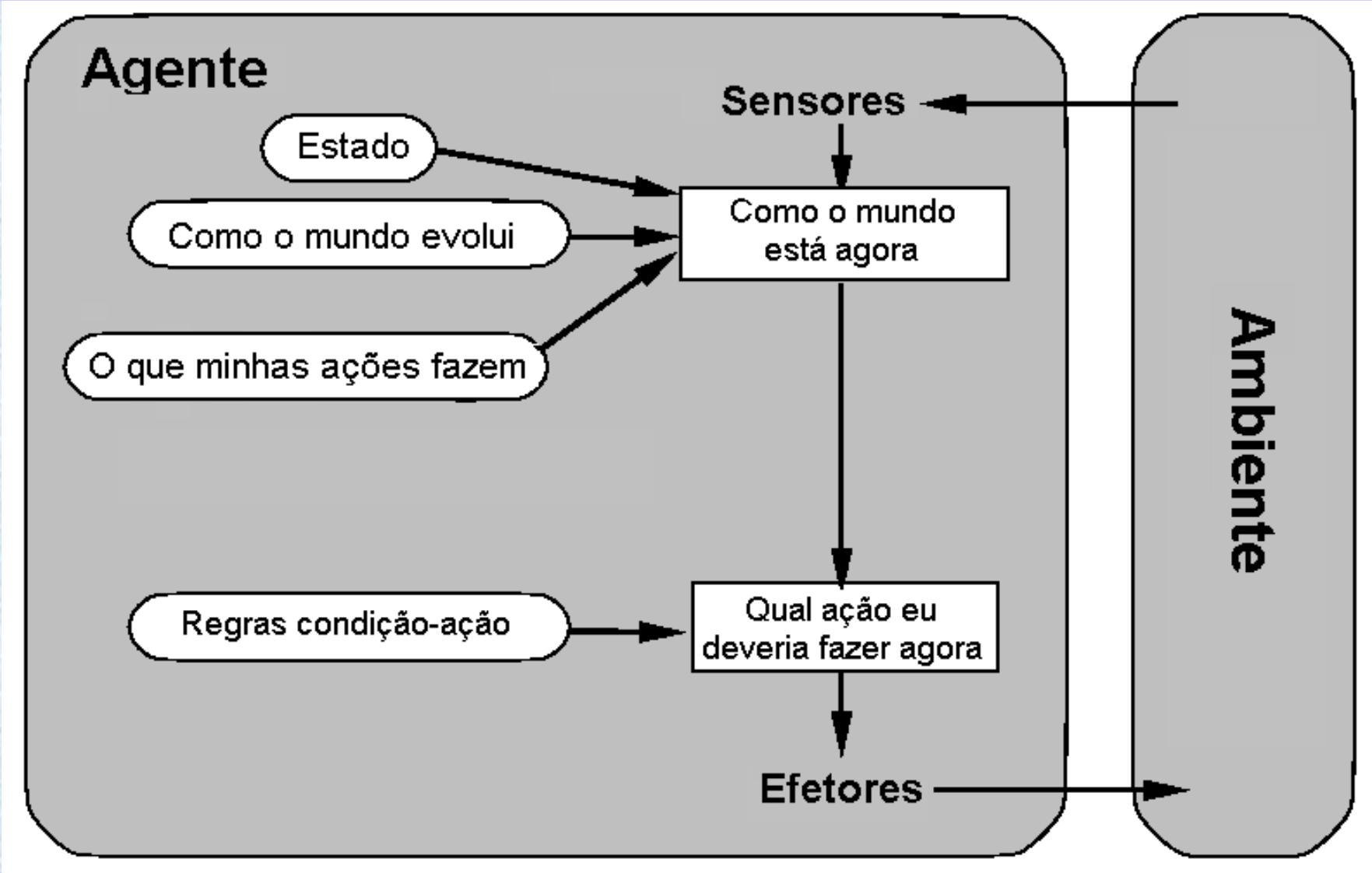
estado ← INTERPRETA_ENTRADA(percepção)

regra ← QUE_REGRA(estado, *regras*)

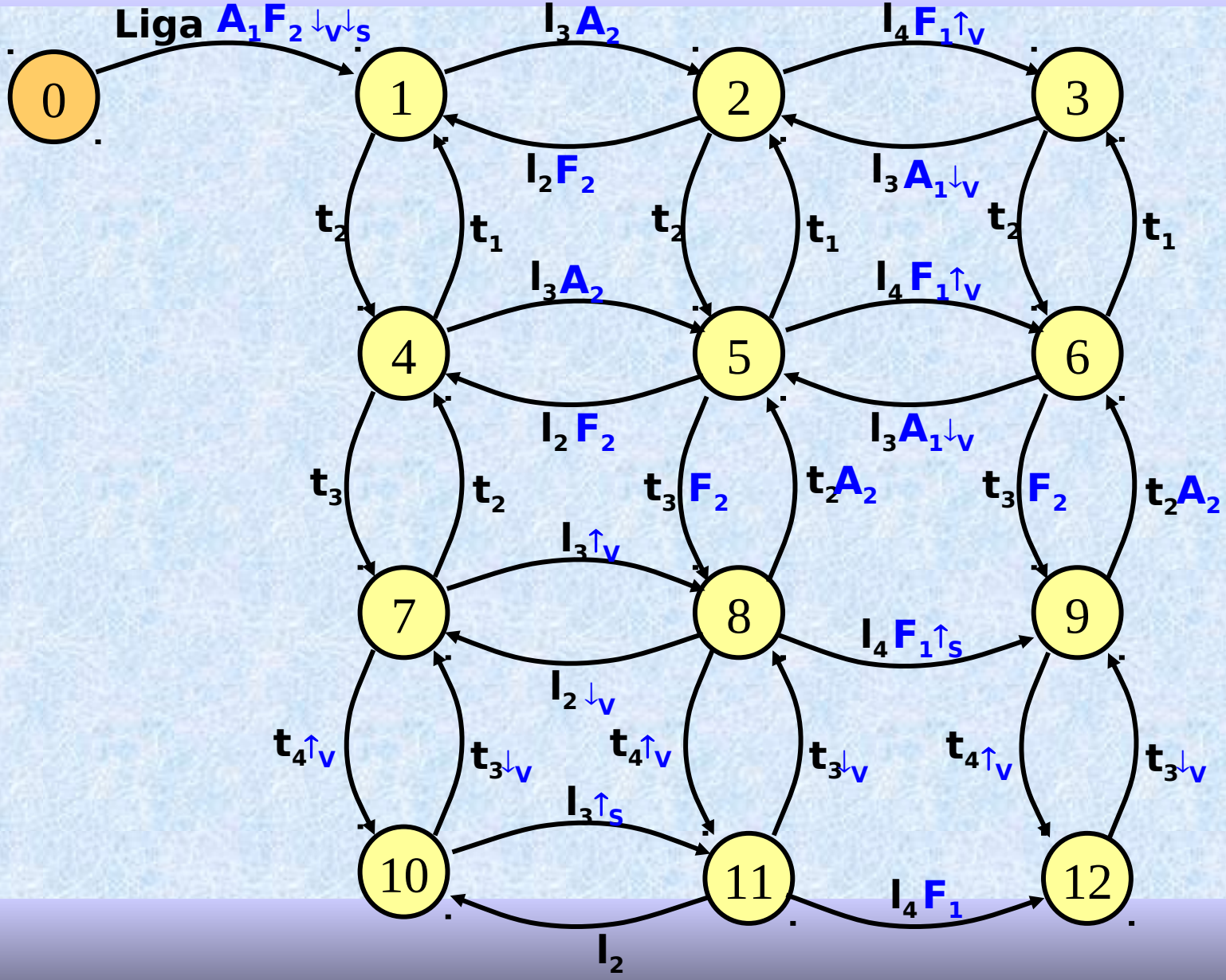
ação ← REGRA-AÇÃO[regra]

return ação

Agente Reativo com Estado (interno)



O exemplo da caixa d'água



O exemplo da caixa d'água

EA	PE	Regra	Ação
0	1	Liga	$A_1 F_2 \downarrow_v \downarrow_s$
1	2	l_3	A_2
1	4	t_2	
2	1	l_2	F_2
2	3	l_4	$F_1 \uparrow_v$
2	5	t_2	
3	2	l_3	$A_1 \downarrow_v$
3	6	t_2	
4	5	l_3	A_2
4	7	t_3	
5	4	l_2	F_2
5	6	l_4	$F_1 \uparrow_v$
5	8	t_3	F_2
6	3	t_1	

EA	PE	Regra	Ação
7	4	t_3	
7	8	l_3	$\uparrow_v$
7	10	t_4	$\uparrow_v$
8	5	t_3	A_2
8	7	l_2	$\downarrow_v$
8	9	l_4	$F_1 \uparrow_s$
8	11	t_4	$\uparrow_v$
9	6	t_3	A_2
9	12	t_4	$\uparrow_v$
10	7	t_4	$\downarrow_v$
10	11	l_3	$\uparrow_s$
11	8	t_3	$\downarrow_v$
11	10	l_2	

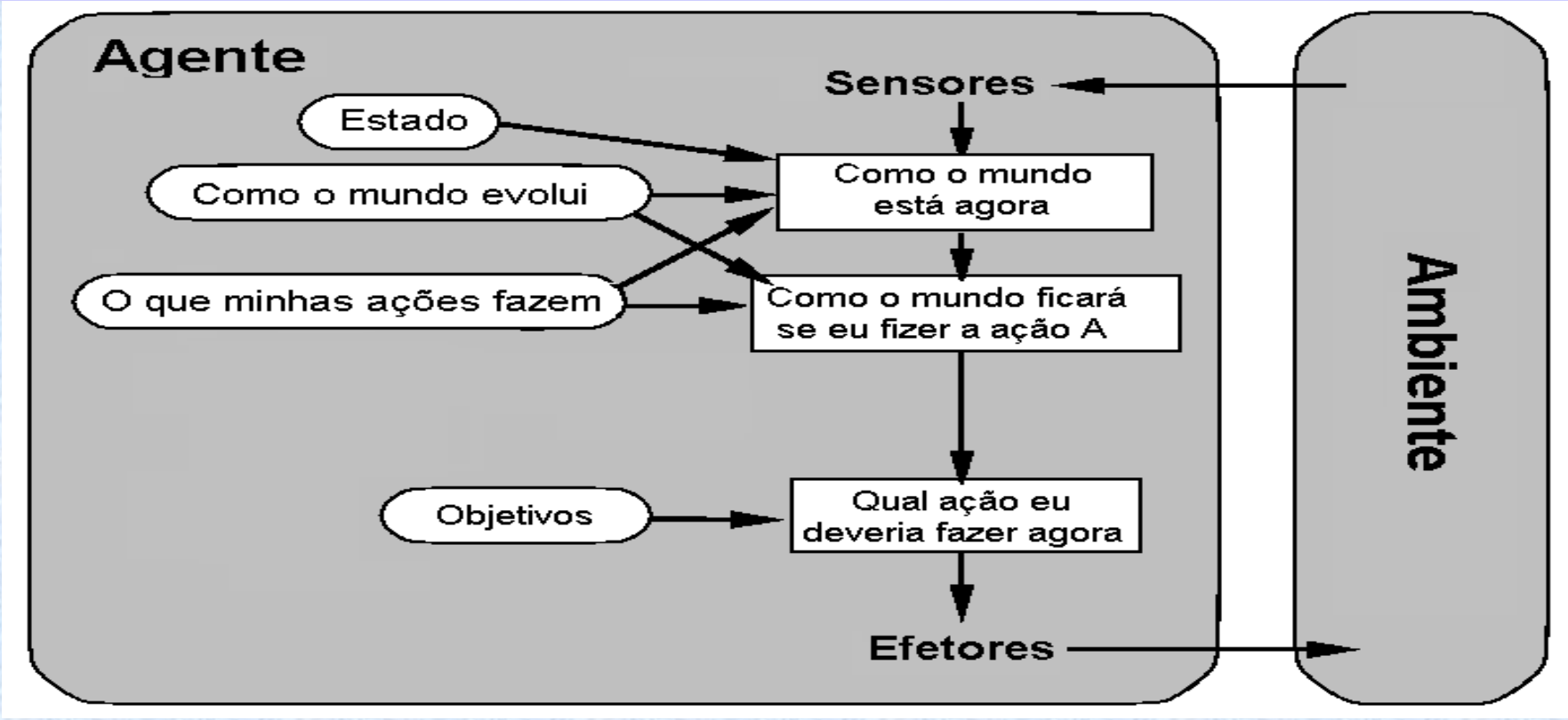
Agentes Reativos com Estados

function AGENTE_REATIVO_COM_ESTADO(percepção): ação
static: estado (uma descrição do estado atual do mundo)
regras (um conjunto regra-ação)
ação, a ação mais recente, inicialmente nenhuma

estado ← ATUALIZA_ESTADO(estado, percepção)
regra ← QUE_REGRA(estado, *regras*)
ação ← REGRA-AÇÃO[regra]
estado ← ATUALIZA_ESTADO(estado, ação)

return ação

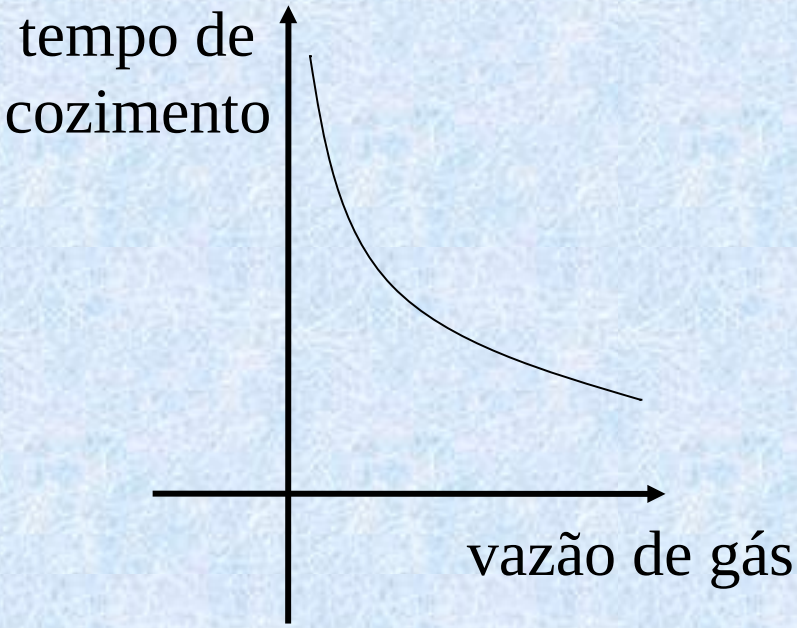
Agentes baseados em Objetivo



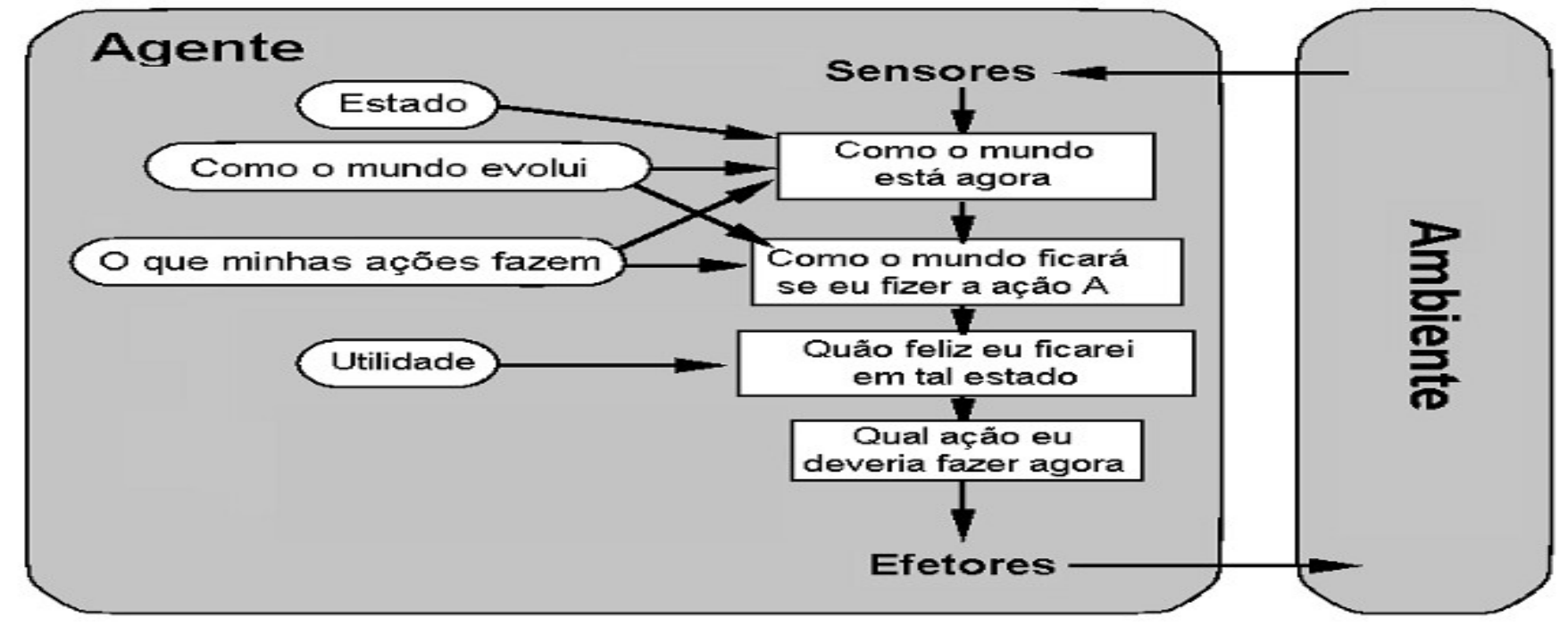
No modelo REATIVO, as informações não são representadas explicitamente, pois existem regras diretas de ação a cada percepção. O táxi freia, quando vê luz de freio a frente. O modelo REATIVO não sabe que se as luzes de freio estão acesas a velocidade do carro à frente diminuirá.

No modelo por OBJETIVO, sabendo que como o mundo evolui, a única ação a ser feita para atingir o objetivo de não bater é frear.

Agentes baseados em Objetivo



Agentes baseados em Utilidade

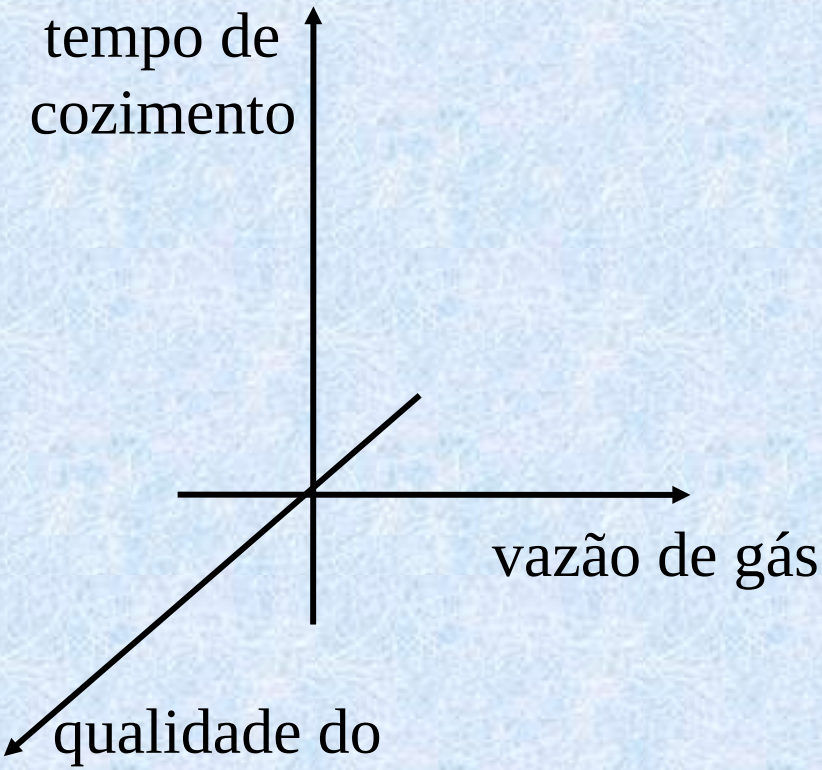


Se um estado do mundo for preterido em detrimento de outro, ele terá mais **utilidade** para o agente.

Uma **função de utilidade** permite decidir entre vários estados, qual o melhor.

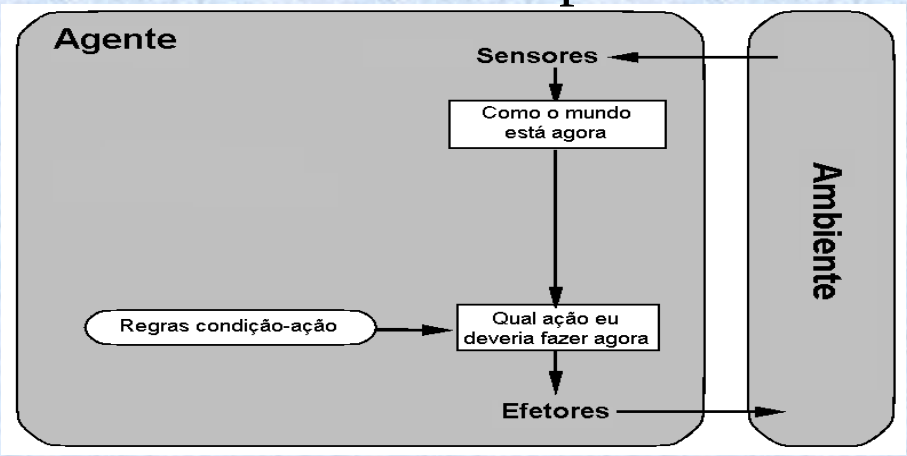
Exemplo: se temos objetivos contraditórios como velocidade e segurança, a função utilidade especifica o compromisso apropriado.

Agentes baseados em Utilidade

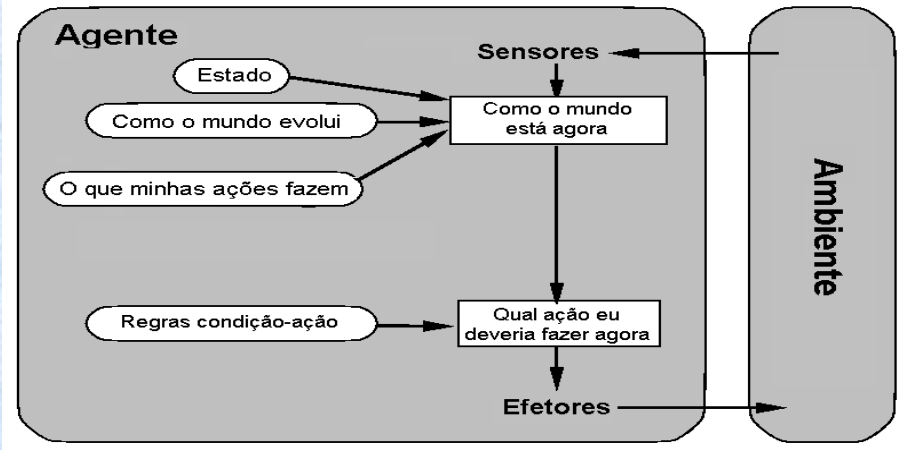


Resumo de modelos de agentes

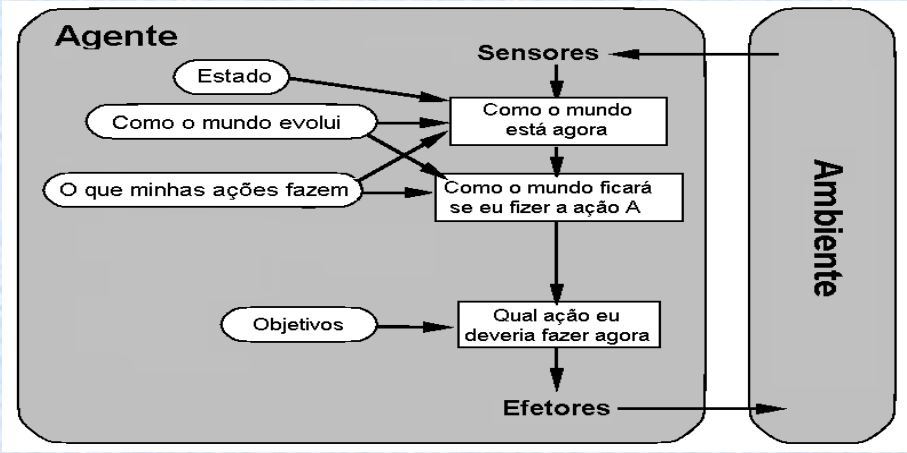
Reativo Simples



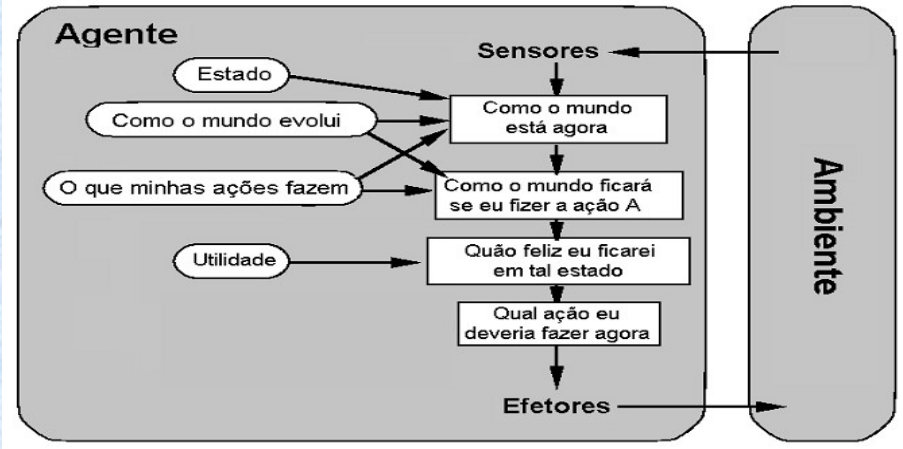
Reativo com Estado



Baseado em Objetivo



Baseado em Utilidade



Agentes baseados em Utilidade

- Objetivos conflitantes
  - refrigerar mancais
  - utilizar água com temperatura variável
- Um conjunto de objetivos
  - manter a temperatura em T
  - não deixar faltar água
  - não deixar extravazar a caixa

Programas Ambientes (sem escores)

```
procedure EXEC_AMBIENTES(estado, FUN_ATU, agentes, término)
    inputs:     estado,          // o estado inicial do ambiente
                FUN_ATU,   // função para modificar o ambiente
                agentes,         // um conjunto de agentes
                término     // uma condição indicando o final
    repeat
      for each agente in agentes do
        agente.percepção ← OBTÉM_PERCEP(agente, estado)
      end;
      for each agente in agentes do
        agente.ação ← agente.PROGRAMA(agente.percepção)
      end;
      estado ← FUN_ATU(ações, agentes, estado)
    until término(estado)
```

Programas Ambientes (com escores)

```
procedure EXEC_AVALIA_AMBIENTES(estado, FUN_ATU, agentes,
    término, ESCORE_FUN): escore
    locals:      pontos       // um vetor do mesmo tamanho de
       agentes
                                // inicialmente com zeros
    repeat
       for each agente in agentes do
         agente.percepção ← OBTÉM_PERCEP(agente, estado)
       end;
       for each agente in agentes do
         agente.ação ← agente.PROGRAMA(agente.percepção)
       end;
       estado ← FUN_ATU(ações, agentes, estado)
       pontos ← ESCORE_FUN(pontos, agentes, estado)
    until término(estado);
    return pontos;
```